max_Synchronization.py and log_Synchronization.py

1. Create three initial conditions i-j, j-i, and abs(i-j) where (i,j) is a coordinate as the initial sample size.
2. Perform LPP or DPRE from start (line) to end (n) traveling as many steps as needed.
3. Plot the saved steps using Matplotlib and create a video.

cupySubprocesses.py and numpyNumbaSubprocesses.py

1. Choose a t_start and t_end along with how many increments you want to save from t_end. You can also set the initial condition.
2. Filenum will create as many processes you want so that you can run multiple processes to generate as many samples as you want. n_samples is how many samples each process will create.
3. It is recommended you choose 1 GPU and CPU Cores equal to the number of processes you want as it was found to be the most efficient.

pathCode.py

1. Performs regular LPP or DPRE, but also finds the geodesics of each point at each time step.
2. The cords function was used to find the intersection points of the coordinates (0,1) and (1,0) (t=19,999) along with (1,-1) and (-1,1) (t=20,000) using argmax from numpy.
3. It was also recorded whether (0,1) or (1,0) is linked to (0,0).
4. The increments between the point (0,0) and points (1,0), (0,1), (1,-1), and (-1,1) were also taken.